International Association of Scientific Innovation and Research (IASIR)
(An Association Unifying the Sciences, Engineering, and Applied Research)

ISSN (Print): 2279-0047
ISSN (Online): 2279-0055

International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)

www.iasir.net

Generation of maps using a Pioneer 2DX mobile robot in a simulated environment Player/Stage

Guillermo Ceme [1], Michel Garcia [2], Cinhtia González[3], Sergio González[4]
Autonomous University of Yucatan.Multidisciplinary Unit Tizimin.
Calle 48 Num. 207 x 31. CP 97700Tizimin, Mexico

Abstract: *This paper describes the implementation of an algorithm that creates maps of an unknown environment for a robot. The algorithm is tested using a model of the Pioneer 2DX mobile robot found in the Player/Stage simulator, and a virtual environment that reflects a real place in the Autonomous University of Yucatan. The tests run simulations of the mobile robot traveling through the virtual environment. A joystick is used to manually control the robot, and a laser sensor is used for the detection of obstacles and walls. Maps obtained using the algorithms are able to represent the environment with the required precision for the robot to navigate.*

Keywords: *Map building, Mobile robot, Simulated environment, Player Stage, Pionner 2DX.*

I. Introduction

Robotics is a science with a wide variety of applications. Its results are applied in areas ranging from investigations to solve small scale problems to complex applications, many of which can be applied in places of difficult access, such as outer space and the depths of the sea. Robots are also used in industry as a manufacturing tool; in the military scope; etc. [1].

In addition, mobile robots are beginning to gain acceptance in many areas of daily life, as they are already employed in housework like vacuuming the house and cleaning the pool [2], what makes necessary for robotic systems to recognize the place where they are.

So that a robot could plan and follow a trajectory, it is very useful having a map on which it could be planed a route [3]. The creation of maps is very helpful for the robot to localize itself and then perform other tasks indicated. Therefore, the construction of maps becomes one of the main objectives to develop truly autonomous mobile robots [4], especially in hazardous works that take place in unknown areas in which it is required to navigate, dodge obstacles and trace routes.

The generation of maps is framed in cartography and frequently focuses specifically on the interpretation of the immediate surroundings of a specific robotic system. In this regard, automated tools are produced increasingly more to support the generation, interpretation, domain and use of the maps, and the proof is the growing GIS (Geographic Information Systems) that are being introduced in the market and in the scientific fields, and are evolving in unexpected ways. To quote some of the most known cases in the civil sector, Google Earth and Maps, ArcGis tools, car navigators, GPS (Global Positioning System), GLONASS, etc. [4]

In this work is implements an algorithm that allows the robot build a map of its environment using a laser sensor which captures the distance between the robot and an object.

II. Methodology

In the development of this Project, it was used the set of libraries Player and its associated simulator Stage [5]. The chosen robot is a Pioneer 2DX [6], which comes incorporated in Stage and can use the laser sensor model LMS200 [7]. It was decided to use that laser because of its accuracy when detecting obstacles.

The environment for mapping is a real terrain of the Autonomous University of Yucatan (UADY), specifically in the Multidisciplinary Unit Tizimin (UMT). The environment is represented by a plane with obstacles where the robot will navigate in order to build an internal map which subsequently will help to achieve a successful navigation.

Player also offers the possibility to interact with the hardware of the robot by using specialized drivers. This communication is achieved by using some standard interfaces included in the software of simulation.

Simulation Player/Stage

In addition to *Player* it was used *Stage*, which is a simulator that works directly with *Player*, so that even in absence of physical equipment, the algorithms can be tested in a simulation with a great similarity to reality. Stage comes with several predefined models of robots and models of devices, some of which are used in this work.

Robot Pioneer 2DX

Stage incorporates the model of the robot *Pioneer 2DX*, which can be observed in Figure 1. The laser sensor *LMS200* (Fig. 2) was incorporated into this model. The real robot has the following features: weights 20 Kg, its maximum speed is 1.6 m/s, and has two differential wheels.

Figure1 Pioneer 2DX Robot.

The laser sensor used to perform the required measurements has a maximum reach of 8 meters and covers an angle of 180°.

Figure 2 LMS200 Laser.

Interfaces and programming

For programming it was used the C++ language, which was linked with *Player* adding the corresponding libraries. The development was performed with C++ [8] because the structure of the language is easily coupled to functions that allow controlling the robot devices. The two used interfaces are: *Position2dProxy* to obtain the position in which the robot is located (coordinates *X* and *Y*, and angle) as well as to control the movement of the robot; and *LaserProxy* to obtain information of the laser device, which measures the distance of the obstacle closest to the robot.

Detection of objects and walls

To calculate the coordinates where the objects closer to the robot are, such as walls and obstacles, the trigonometric ratios *sine* and *cosine* are used. Using the distance *d* and angle *a* obtained from the laser sensor when an object is detected, *d* is considered as the hypotenuse of a right-angled triangle, so that the position of the detected object is calculated using (1) for the *X axis* and using (2) for the *Y axis*.

$distX = d + cosine(a)$ (1)

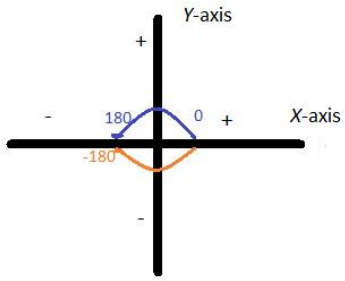
$distY = d + sine(a)$ (2)

The obtained values, *distX* and *distY*, are the distances of the object from the current position of the robot on their respective axis.

The map is build considering a Cartesian plane in which the initial position of the robot is the origin in the plane *(0,0)*, and as the robot moves, its position is updated taking into account that in the simulation a movement to the right produces a positive increment on the *X*axis, and an upwards movement produces a positive increment on the *Y* axis. Therefore moving towards the left decrements the coordinate on the *X* axis, as well as and moving down decrements the coordinate on the *Y* axis.

For each movement of the robot, its rotational angle is obtained. If the robot is oriented exactly to the right, its angle is 0°. Counterclockwise turns produce a positive increment in the value of the angle, up to 180°; therefore, clockwise turns produce a decrement on the angle, to a minimum of -180°. Thus, the robot can turn onto itself and the angle will always be between 180° and -180° inclusive. Fig 3 illustrates the explanation above.

Figure3 Position and orientation of the robot.



The simulated environment is based on a sketch of the Computing Centre of the Multidisciplinary Unit Tizimin; Figure 4 shows the original sketch.

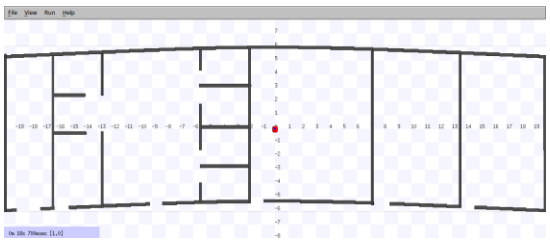
Figure4 Originalsketch.

III. Development

The *Player/Stage*simulation environment was used to implement the mapping algorithm due to its robustness and clarity of execution. At the beginning it is necessary to provide the simulator *Stage*with an image of the land that will be used for the navigation of the robot, as well as to define the robot to use. In this case the *Pioneer 2DX* is already defined in the *Player*'s libraries so it is only selected.

The simulation is executed from a terminal (command "*player empty.cfg*"). In Figure 5 is presented that simulation executed with the sketch previously adapted to perform the tests.

Figure 5 Map of the UMT in the simulation of Stage



To control the movements of the robot, it is required to run the Joystick library from another terminal using the command "*playerjoy*".

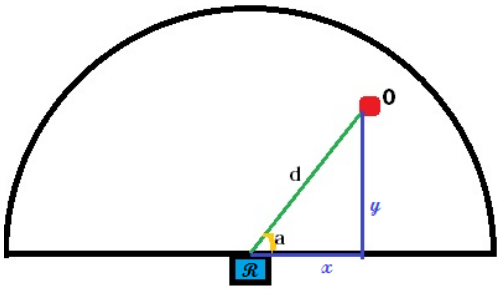
Once started the simulation, it is necessary to run the mapping algorithm, for which another terminal is opened to execute the command "*./program*". In this case the software throws coordinates in the form of 2 columns. To save the produced data in a text file it is added to the above command an option more, becoming as follows: "*./programa> data*".

A. Mapping algorithm.

The algorithm which detects obstacles and walls is based on the calculation of coordinates using trigonometric ratios, taking into account that a right-angled triangle is generated whose hypotenuse is the measured distance, its angle 'a' is also measured, and its legs (sides) are the unknown rectangular coordinates relative to the robot, as shown in Fig. 6.

The laser sensor has a range of 180 degrees, which is divided into 360 points. For each point is obtained a measurement of the distance which reaches. If it detects any distance less than 8 meters (maximum distance that the sensor detects), the presence of an object is considered and a point is marked in the map that is being generated. Figure 6 is a scheme of the laser sensor function.

Figure1 The measurements of the laser sensor are: angle (a) and distance (d), from the robot (R) when detects an object (O).



To obtain the coordinates of each point relatives to the robot, where the relative coordinates *x* and *y* are the distance of the obstacle on the x-axis and y-axis respectively, they are calculated using the angle and distance provided by the laser sensor, as shown in Figure 6.

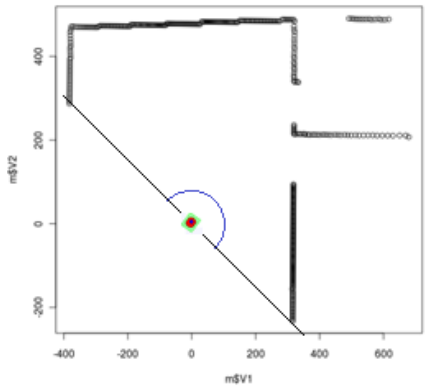
To obtain the exact position of the object in the map, it is added the current position of the robot to the previously calculated coordinates, i.e., the distance of the obstacle on the *x-axis* is added with the value on the *x-axis* of the robot's position, and in the same way, the calculated distance of the obstacle on the *y-axis* is added with the value on the same *y-axis* of the robot's position, thus obtaining the absolute coordinates of the detected object.

The duration of the implemented algorithm can be controlled so that the process last from a few seconds to several minutes, which allows to define the map accuracy to a desired level. While more time is invested in the algorithm, better will be the quality of the generated map.

IV. Results

In tests performed with the robot in a fixed position, the algorithm shows approximately a 90% in accuracy, as can be seen en Figure 7, which shows the generated map with the scope of the 180° visible to the robot's laser sensor.

Figure 2 Map generated using a fixed position



When the robot moves on straight line, the algorithm manages to represent the plane with a precision greater than 90%, thus generating a better approximation of the environment than being stationary, as shown in Fig. 8, where the start and end points of the robot's journey are deployed. Figure 9 is the reconstruction of the environment captured by the robot.

Figure3 A straight line trajectory, from the start point (i) to the end (f).

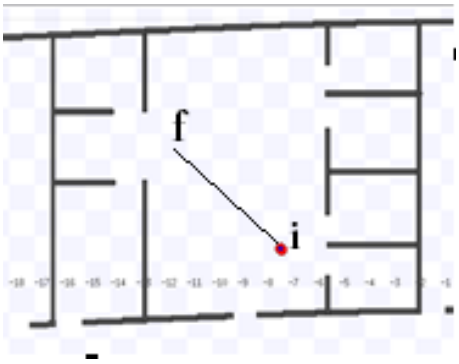
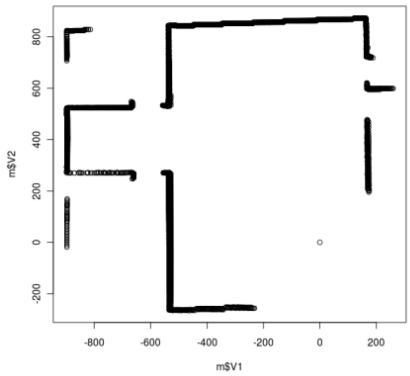


Figure4 Map generated during the straight line displacement (the one shown in Figure 8).



When the robot performs turns during its displacement, the program produces inaccurate data. It is thought to be because the *Joystick* tool runs in a different process of the main program, which causes that some noise is generated when the program register the lectures of data. The results of a smooth turn can be seen in Figure 10, and Figure 11 shows the map generated with a pronounced turn.

Figure5 Map constructed in a curved path with a smooth angle.

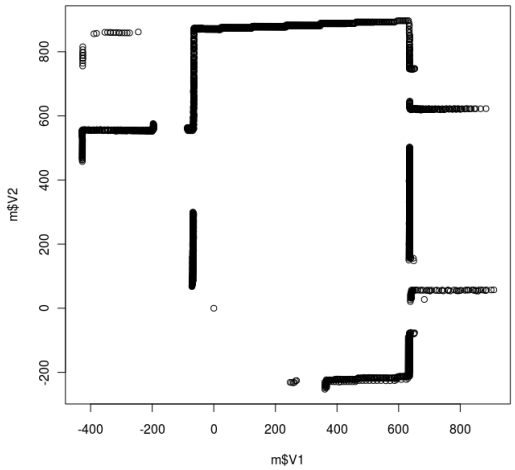
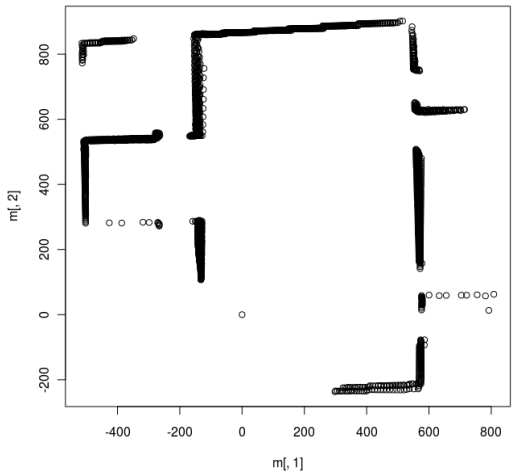


Figure6 Map constructed in a curved path with a more pronounced angle.



Therefore, the implemented algorithm requires exploring the terrain making moves in a straight line, for greater precision in measurements.

V. Conclutions

The mapping algorithm presented in this work allows obtaining images with more than 90% of precision, when the environment is traveled in straight line. When the robot performs curved paths, the generated map is of lower quality, however, in both cases it is achieved the goal of obtaining information with enough precision to build maps that represents the environment in such a way that can be used for planning routes and/or exploration.

Finally, as part of the future work it is intended to improve the algorithm in the specific case of the turns, as well as the integration of an algorithm that allows the robot to move autonomously while mapping the terrain.

VI. References

[1] Rebollo Balaguer, Tomás. (2010). Diseño e implementación de una capa de software para el control de robots mediante Player-Stage, E-Archivo, el Repositorio Institucional de la Universidad Carlos III. Recuperada en Mayo 9, 2012, del sitio Web temoa : Portal de Recursos Educativos Abiertos (REA) en http://www.temoa.info/es/node/204347

[2] J. M. Angulo, S. Romero and I. Angulo, "Introducción a la robótica", 1ª ed., Ed Thomson. Spain. 2005.

[3] E. Mariscal, "Planeación y Seguimiento de Trayectorias de Robots Móviles en una Simulación de un Ambiente Real". Ra Ximhai. Vol 1. Number 1, Enero-Abril 2005. pp. 177-200. Available: http://www.ejournal.unam.mx/rxm/vol01-01/RXM001000112.pdf

[4] D. González. "Generación automática de mapas en espacios cerrados mediante robots móviles". Graduated Tesis of Ingeniería Informática y de Telecomunicación, Escuela Politécnica Superior. Universidad Autónoma de Madrid. September, 2011.

[5] Player/Stage website. (2013) Available: http://playerstage.sourceforge.net/.

[6] (2013) "Activmedia Robotics PIONEER 2 DX". [Online]. Available: http://www-lar.deis.unibo.it/equipments/p2dx/index.html

[7] (2013) "Interfacing with the Sick LMS-200". [Online]. Available: http://www.pages.drexel.edu/~kws23/tutorials/sick/sick.html

[8] (2013) libplayerc++ Client libraries. [Online]. Available: http://playerstage.sourceforge.net/doc/Player-2.0.0/player/group__player__clientlib__cplusplus.html

VII. Acknowledgments

The authors thankfully acknowledge to the Autonomous University of Yucatan for their support in the publication of this paper.